

# MATLAB EXPO

## MATLAB을 활용한 간편 데이터 분석 및 인공지능

장규환 부장(Ph.D), 매스웍스코리아



# What are “low code” tools?

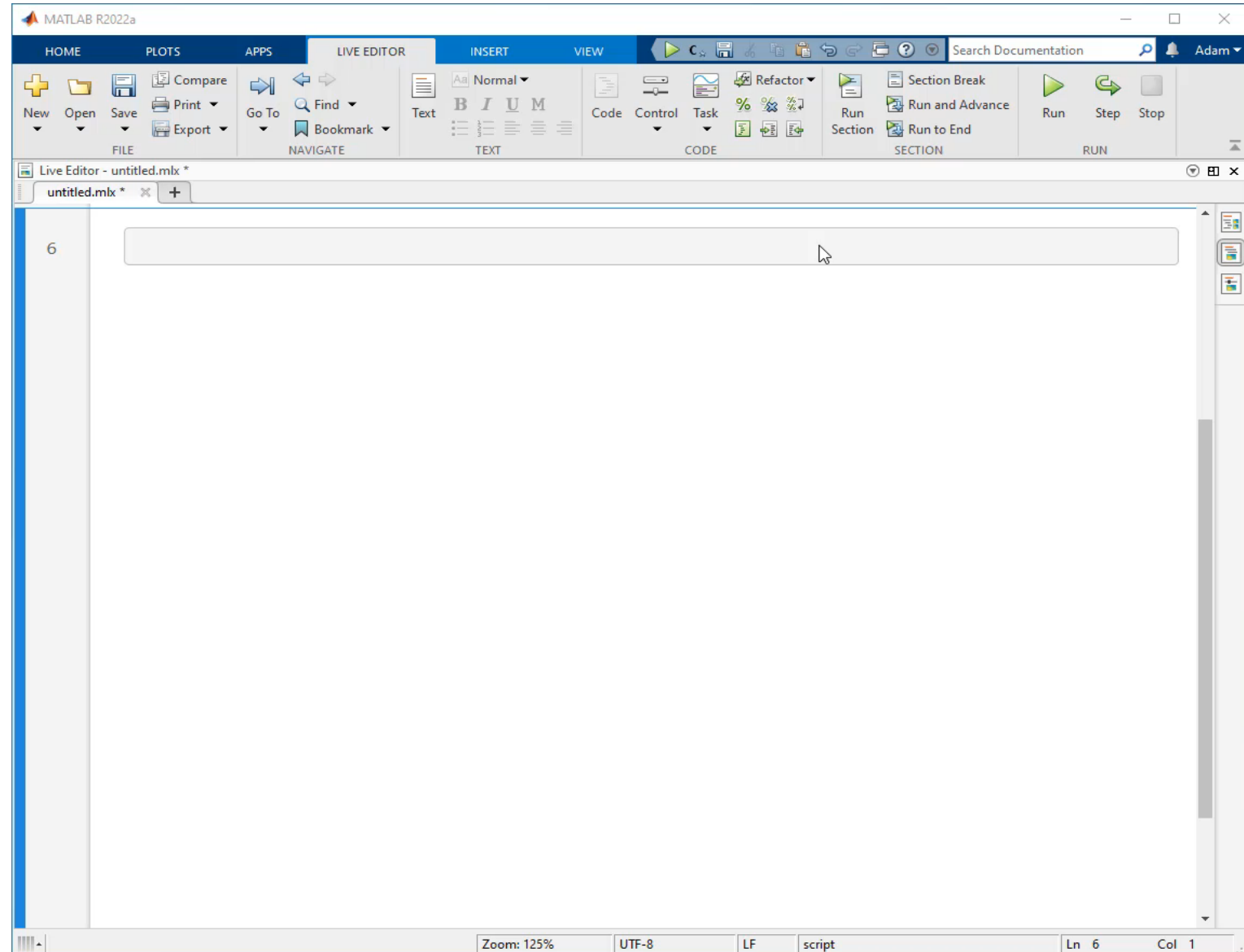
**Low code** tools enable:

- rapid software development
- minimal manual coding

## Benefits of low code tools:

- Shallow learning curve
- Teach *how* to code
- Solve task first, code later

***Not just for beginners***



# What is the Live Editor?

- See your results alongside the code that produced them
- Run sections of code individually or run the whole file
- Find errors at the location in the file where they occur

The screenshot displays the MATLAB Live Editor window. The top part shows a scatter plot titled 'Percent Fatalities per Million Vehic' (partially visible) with 'urbanPopulation' on the x-axis (ranging from 20 to 100) and 'Percent Fatalities per Million Vehic' on the y-axis (ranging from 0.005 to 0.02). The plot contains blue circular data points and a red linear regression line.

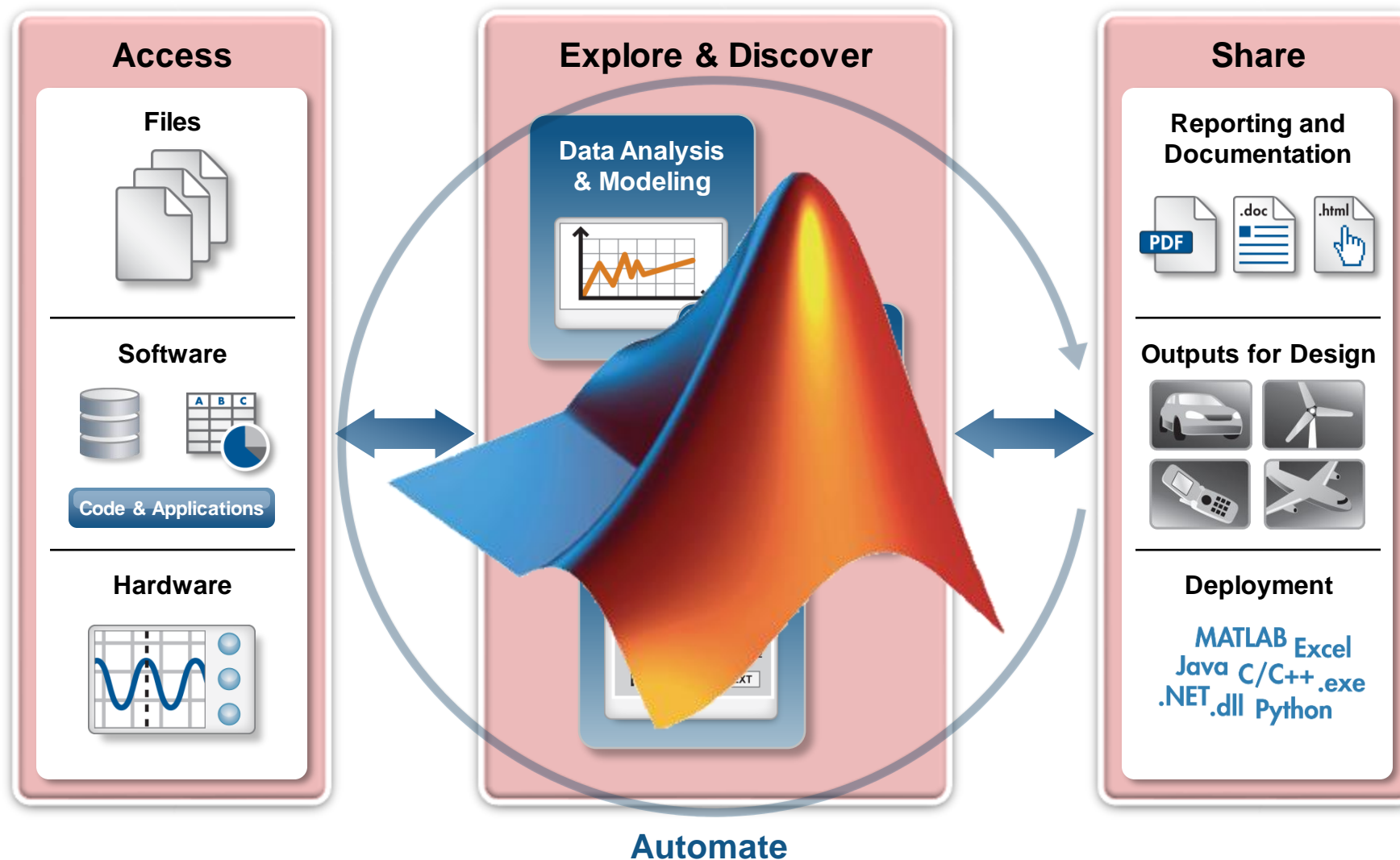
Below the plot, there is a text block titled 'Plot Fatalities and Urbanization on a US Map'. The text reads: 'Based on this analysis, we can summarize our findings using a plot of fatality rates and urban population on a map of the continental United States.'

Underneath the text is a code editor window showing MATLAB code. The first line is `load usastates.mar`, which has triggered an error: 'Error using load Unable to read file 'usastates.mar'. No such file or directory.' The rest of the code is as follows:

```
figure
for i = 1:49
    patch(usastates(i).Lon, usastates(i).Lat, 'white')
end
daspect([1.4 1 1])
axis tight off
hold on
scatter(fatalities.longitude, fatalities.latitude, 2000*rate,...
        fatalities.urbanPopulation, 'filled')
c = colorbar;
title(c, 'Percent Urban')
```

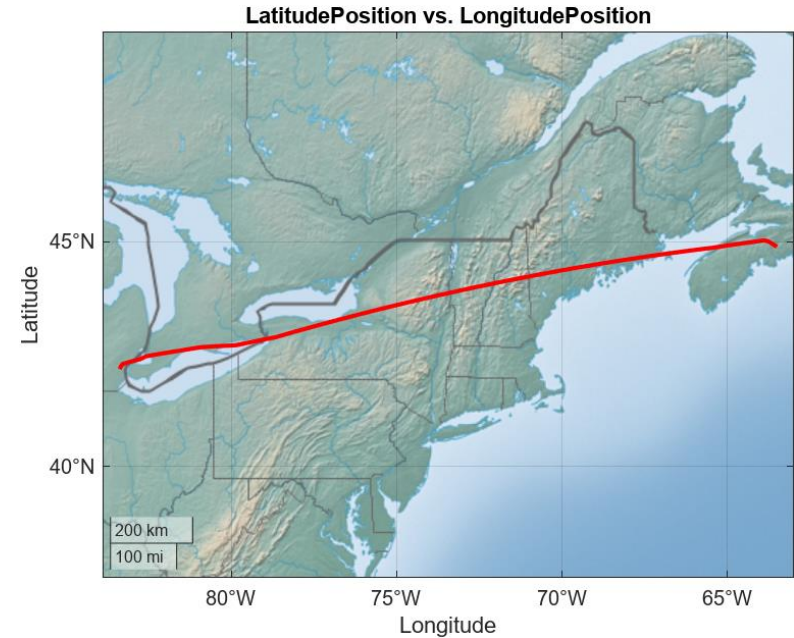
The Live Editor provides a new way to create, edit, run, and share MATLAB code.

# MATLAB simplifies the data analysis workflow with low code tools



# Case Study: Get Flight Sensor Data ready for Modeling

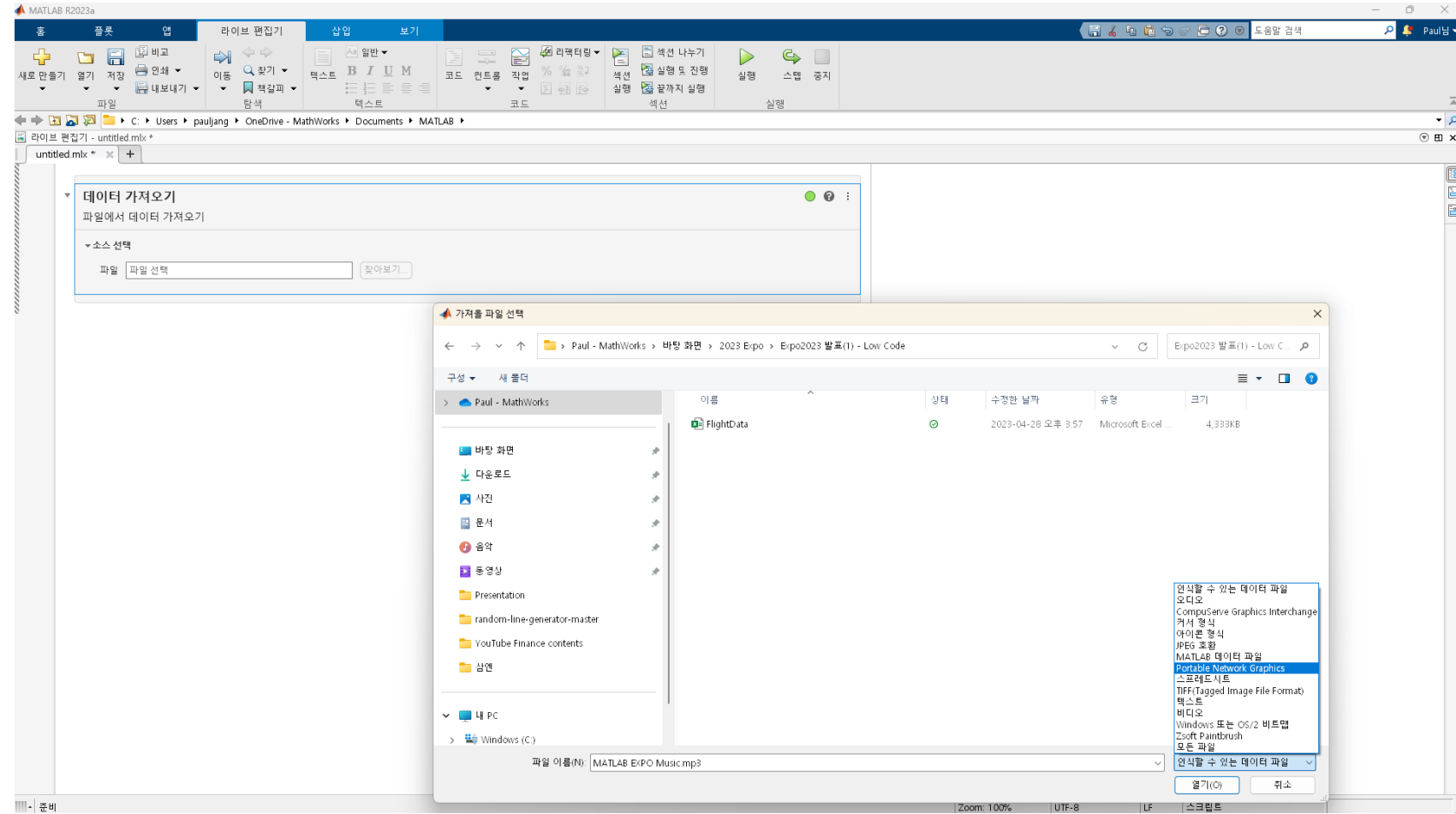
- **Objective:**
  - Explore, analyze and prepare flight sensor data for modeling
  
- **Inputs:**
  - Excel file with raw flight sensor data
  
- **Output:**
  - Cleaned sensor data that can be trained to predict Air Speed
  - Reusable code
  
- **Source:**
  - [NASA Dash Link: Sample Flight Data](#)



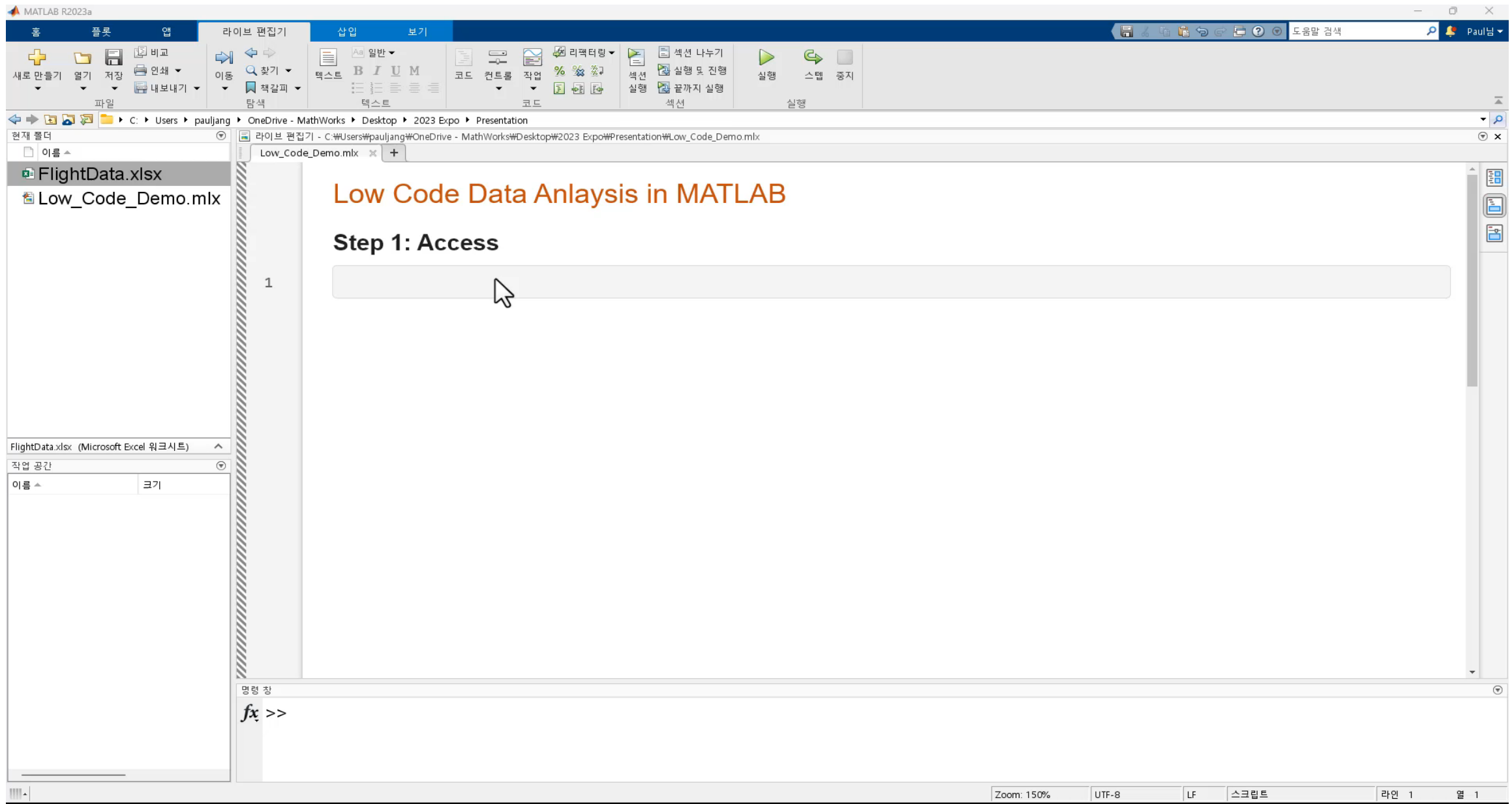
|    | A             | B            | C          | D          | E           | F              | G                   | H              | I        | J                     | K        | L            | M             |
|----|---------------|--------------|------------|------------|-------------|----------------|---------------------|----------------|----------|-----------------------|----------|--------------|---------------|
| 1  | Time          | FuelQuantity | Latitude   | Longitude  | OilPressure | OilTemperature | FlightPhaseFromACMS | WeightOnWheels | Altitude | ExhaustGasTemperature | FuelFlow | TrueAirSpeed | WindDirection |
| 2  | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.519183 | 0           | 23.67477417    | Planning            | 0              | 419      | 17.5                  | 0        | 0            | 0             |
| 3  | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.519183 | 0           | 23.67477417    | Planning            | 0              | 419      | 17.5                  | 0        | 0            | 0             |
| 4  | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.519183 | 0           | 23.67477417    | Planning            | 0              | 420      | 17.5                  | 0        | 0            | 0             |
| 5  | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.519183 | 0           | 23.67477417    | Planning            | 0              | 419      | 17.5                  | 0        | 0            | 0             |
| 6  | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.519183 | 0           | 23.67477417    | Planning            | 0              | 419      | 17.5                  | 0        | 0            | 0             |
| 7  | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.519183 | 0           | 23.67477417    | Planning            | 0              | 420      | 17.5                  | 0        | 0            | 0             |
| 8  | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.519183 | 0           | 23.67477417    | Planning            | 0              | 419      | 17.5                  | 0        | 0            | 0             |
| 9  | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.519183 | 0           | 23.67477417    | Planning            | 0              | 419      | 17.5                  | 0        | 0            | 0             |
| 10 | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.518992 | 0           | 23.67477417    | Planning            | 0              | 419      | 17.5                  | 0        | 0            | 0             |
| 11 | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.518992 | 0           | 23.67477417    | Planning            | 0              | 418      | 17.5                  | 0        | 0            | 0             |
| 12 | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.518992 | 0           | 23.67477417    | Planning            | 0              | 420      | 17.5                  | 0        | 0            | 0             |
| 13 | 6/2/2001 5:41 | 8048         | 44.8915135 | -63.518992 | 0           | 23.67477417    | Planning            | 0              | 419      | 17.5                  | 0        | 0            | 0             |
| 14 | 6/2/2001 5:41 | 8040         | 44.8915135 | -63.518992 | 0           | 23.67477417    | Planning            | 0              | 419      | 17                    | 0        | 0            | 0             |
| 15 | 6/2/2001 5:41 | 8040         | 44.8915135 | -63.518992 | 0           | 23.67477417    | Planning            | 0              | 419      | 17                    | 0        | 0            | 0             |
| 16 | 6/2/2001 5:41 | 8040         | 44.8915135 | -63.518992 | 0           | 23.67477417    | Planning            | 0              | 418      | 17                    | 0        | 0            | 0             |
| 17 | 6/2/2001 5:41 | 8040         | 44.8915135 | -63.518992 | 0           | 23.67477417    | Planning            | 0              | 419      | 17                    | 0        | 0            | 0             |
| 18 | 6/2/2001 5:41 | 8032         | 44.8915135 | -63.518992 | 0           | 25.0178833     | Planning            | 0              | 418      | 17                    | 0        | 0            | 0             |
| 19 | 6/2/2001 5:41 | 8032         | 44.8915135 | -63.518992 | 0           | 25.0178833     | Planning            | 0              | 418      | 17                    | 0        | 0            | 0             |

# Data Access

- Data Import Tool
- Live Script Task
- Various data formats
- Automatic MATLAB code generation
- Distinguishable by sheet in Excel data



# Data Access



The screenshot displays the MATLAB R2023a environment. The main window shows a presentation slide with the following content:

## Low Code Data Analysis in MATLAB

### Step 1: Access

1

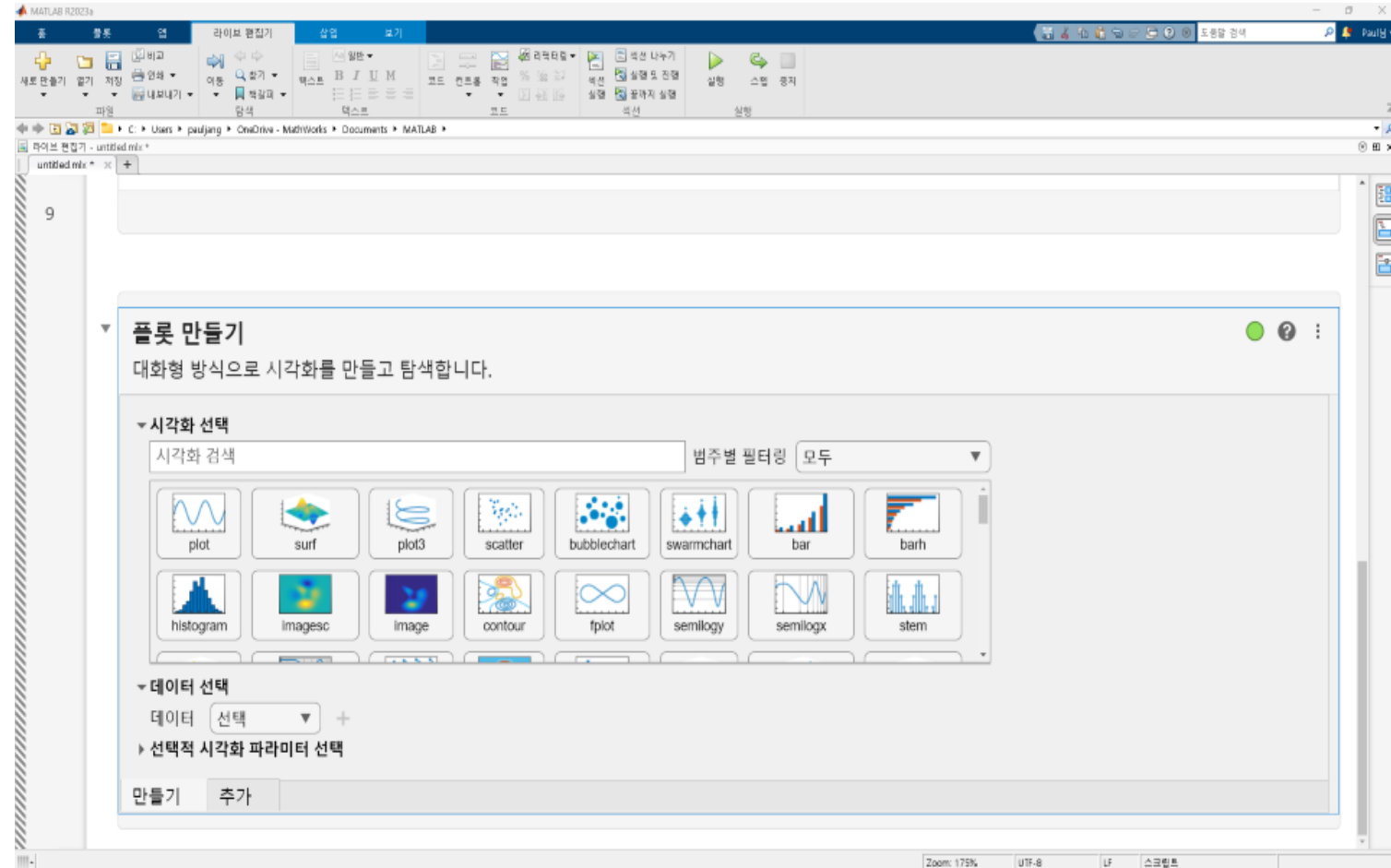
A mouse cursor is positioned over a grey rectangular box below the slide content.

The bottom of the interface shows the Command Window with the prompt `fx >>`.

The interface includes a top menu bar with options like '홈', '플롯', '앱', '라이브 편집기', '삽입', and '보기'. The left sidebar shows a file explorer with 'FlightData.xlsx' and 'Low\_Code\_Demo.mlx'. The bottom status bar indicates 'Zoom: 150%', 'UTF-8', 'LF', '스크립트', '라인 1', and '열 1'.

# Data Visualization

- Live Script Task
- Supporting visualization function suitable for various data
- Automatic MATLAB code generation





# Data Visualization

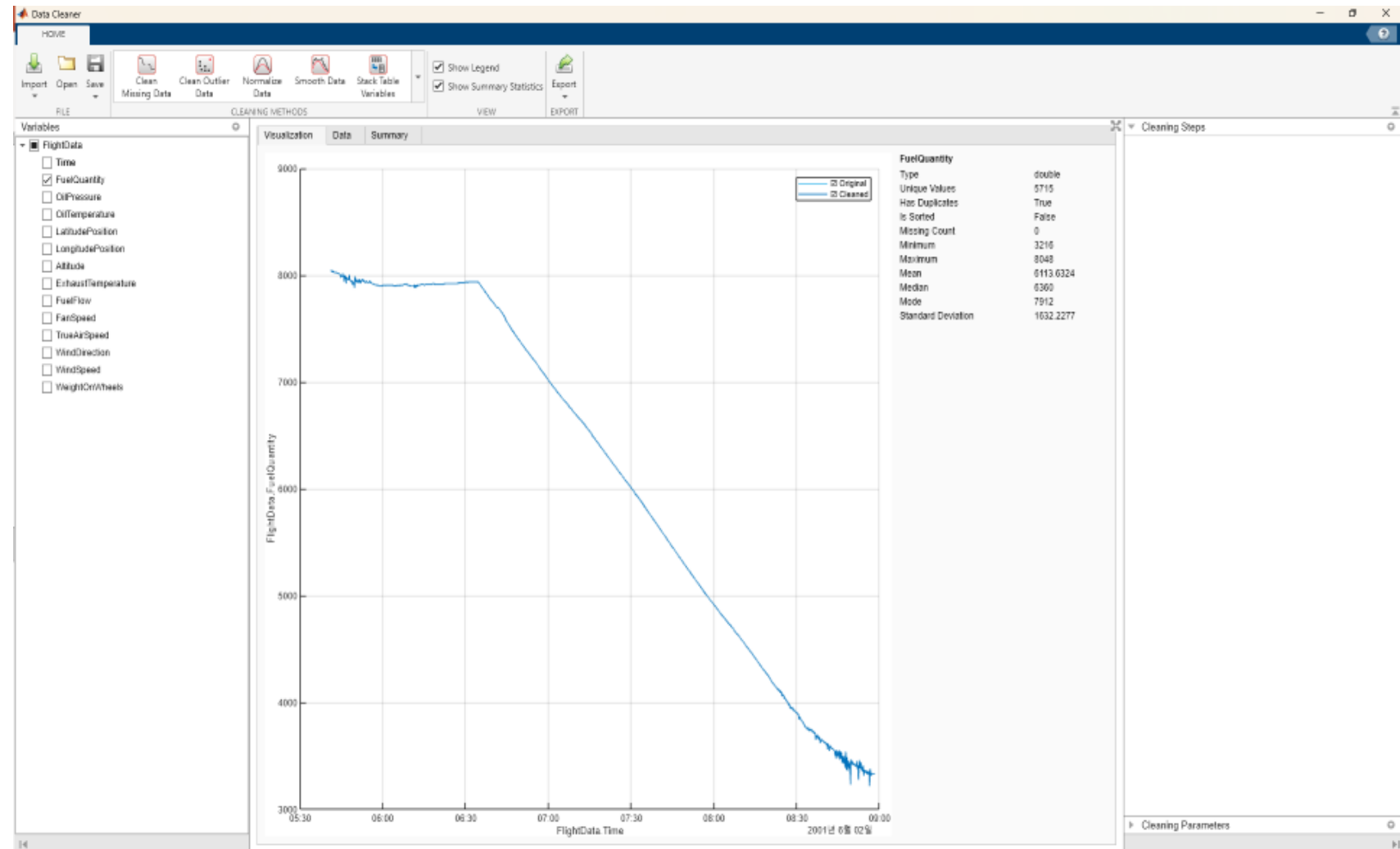
The screenshot displays the MATLAB R2023a environment. The main window shows a presentation slide titled "Step 2: Visualize" with a large empty rectangular area below the title. The left sidebar contains a file explorer showing "FlightData.xlsx" and "Low\_Code\_Demo.mlx". Below the file explorer is a table with the following data:

| 이름         | 크기       |
|------------|----------|
| FlightData | 47312x12 |

At the bottom of the interface, the Command Window shows the prompt `fx >>`. The status bar at the very bottom indicates "Zoom: 150%", "UTF-8", "LF", "스크립트", "라인 23", and "열 1".

# Data Clean

- Data Cleaner App
  - Clean Missing data
  - Clean Outlier data
  - Normalize Data
  - Smooth Data
  - Retime Timetable
  
- Live Script Task
  
- Automatic MATLAB code generation



# Data Clean

The image shows the MATLAB R2023a environment. The main window displays a presentation slide titled "Step3: Clean" on line 31. The variable browser on the left shows two variables: "FlightData" (47312x12) and "h" (1x1). The command window at the bottom shows the prompt `fx >>`.

# Easy access to files, databases, and hardware

- Import Tool and Import Live Task
  - Text, CSV, and Excel files

- Database Explorer (*Database Toolbox*)
  - ODBC
  - JDBC

- Measurement hardware and industrial data
  - Data acquisition hardware (*Data Acquisition Toolbox*)
  - Stand-alone instruments and hardware (*Instrument Control Toolbox*)
  - OPC UA and Aveva PI Server, Modbus devices (*Industrial Communication Toolbox*)
  - CAN, J1939, and XCP (*Vehicle Network Toolbox*)

The screenshot displays the MATLAB software interface. At the top, a workflow diagram shows 'Access' (Files, Software, Hardware) leading to 'Explore & Discover' (Data Analysis & Modeling, Algorithm Development, Application Development) and finally 'Share' (Reporting and Documentation, Outputs for Design, Deployment). Below this, the 'Import' tool is open, showing a table of data with columns 'Time', 'FuelQuantity', and 'OilPressure'. The 'Database Explorer' window is also visible. The main workspace shows a grid of hardware and industrial data toolboxes under the heading 'TEST AND MEASUREMENT':

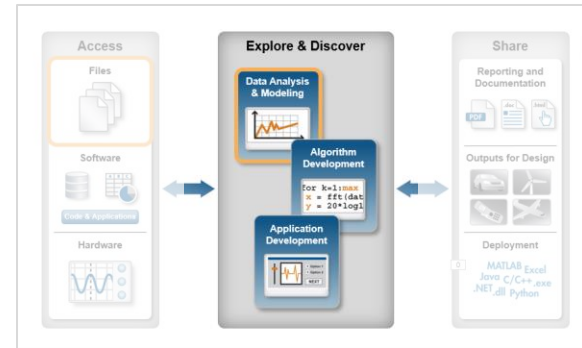
- Analog Input Recorder
- Analog Output Generator
- Arduino Explorer
- CAN Explorer
- CAN FD Explorer
- Embedded Linux Explorer
- Hardware Manager
- Instrument Control
- Modbus Explorer
- OPC Data Access Explo...
- Raspberry Pi Resource M...
- Serial Explorer
- TCP/IP Explorer
- UDP Explorer
- VISA Explorer

At the bottom, a status bar shows 'inventorytable' and 'producttable' with a timestamp of '2012-03-14 13:13:09'.

# Over 100 low code tools for data analysis, application, and AI

- Data Analysis

- Visualize, manipulate, and preprocess
- Math, statistics, and optimization



## Optimize

Minimize a function with or without constraints

### Specify problem type

Objective

Linear

Quadratic

Least squares

Nonlinear

Nonsmooth

Select an objective type to see example functions

Unconstrained

Lower bounds

Upper bounds

Linear inequality

### Constraints

Linear equality

Second-order cone

Nonlinear

Integer

Select constraint types to see example formulas

Solver fminsearch - Unconstrained derivative-free nonlinear minimization (recommended) ?

### Select problem data

Objective function From file Browse... New... ?

Initial point (x0) select ▼

### Specify solver options

### Display progress

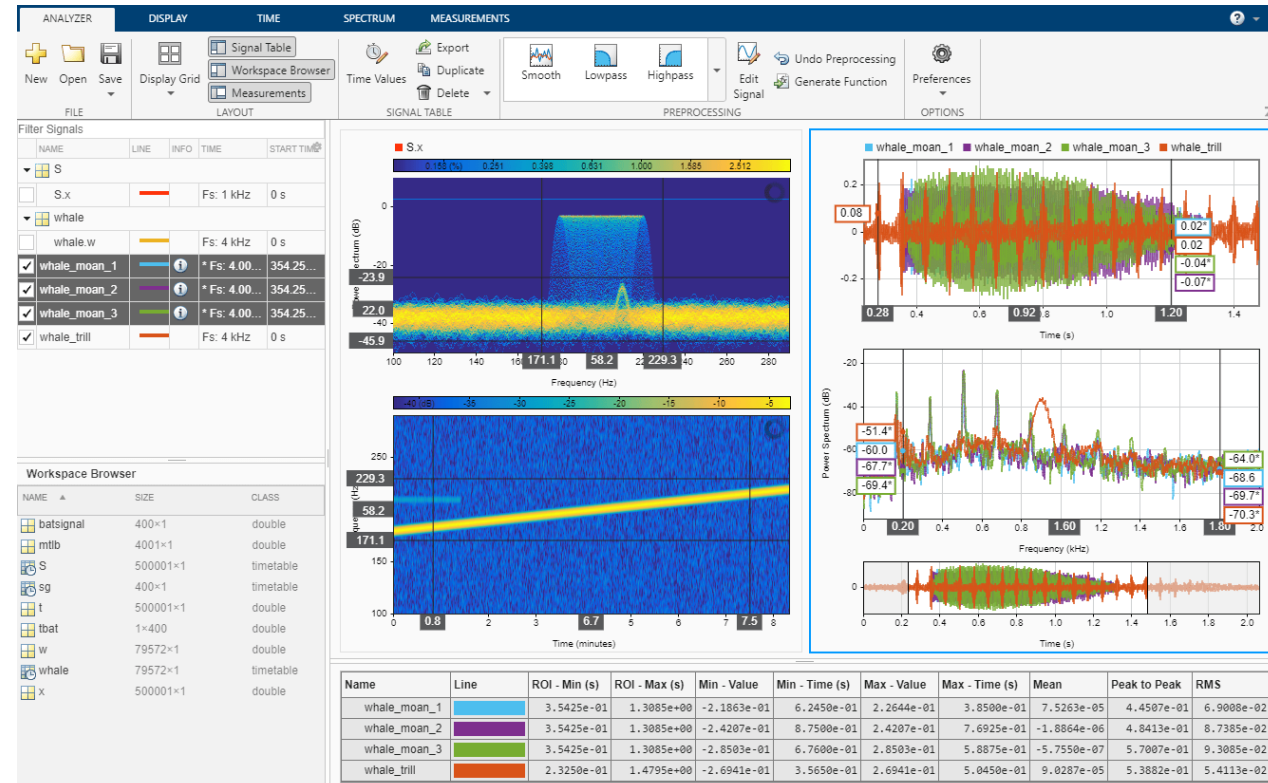
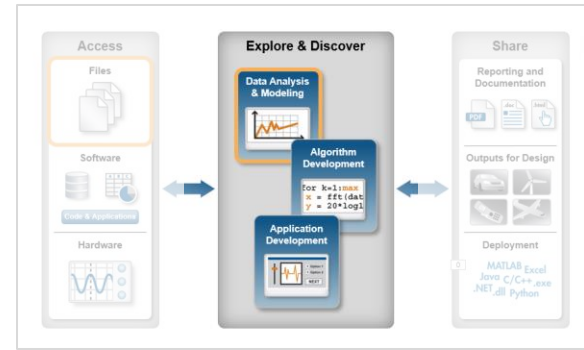
Text display Final output ▼

Plot  Current point  Evaluation count  Objective value

Optimize Live Task (Optimization Toolbox)

# Over 100 low code tools for data analysis, application, and AI

- Data Analysis
  - Visualize, manipulate, and preprocess
  - Math, statistics, and optimization
  
- Application
  - Control system design and analysis
  - Signal processing and communications
  - Image processing and computer vision



Signal Analyzer (Signal Processing Toolbox)

# Over 100 low code tools for data analysis, application, and AI

- Data Analysis

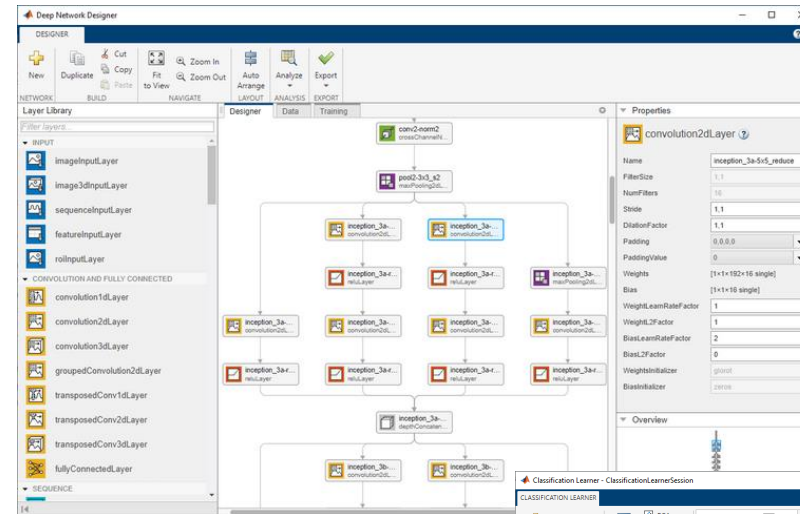
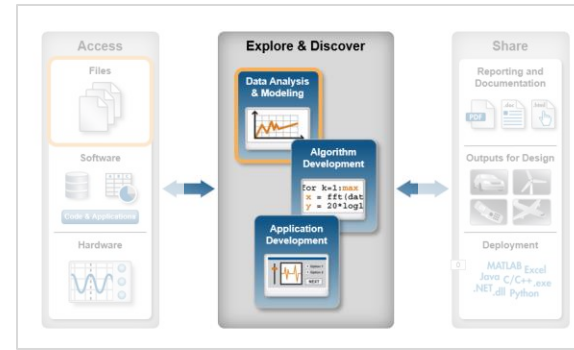
- Visualize, manipulate, and preprocess
- Math, statistics, and optimization

- Application

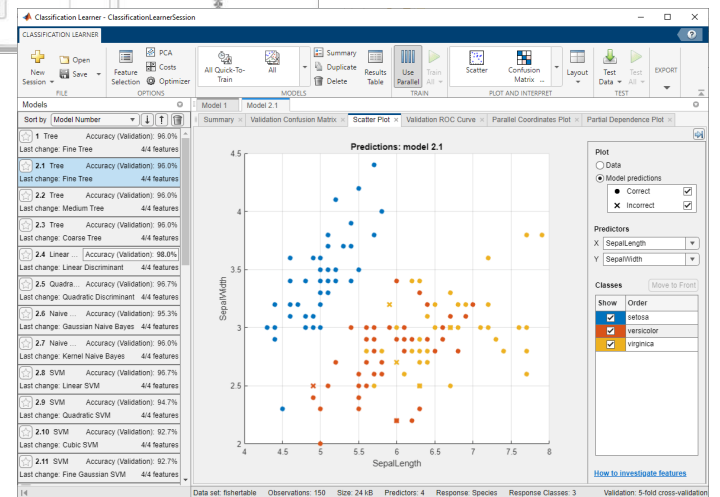
- Control system design and analysis
- Signal processing and communications
- Image processing and computer vision

- Artificial Intelligence

- Ground truth labeling
- Network design, training, and validation
- Quantization and deployment



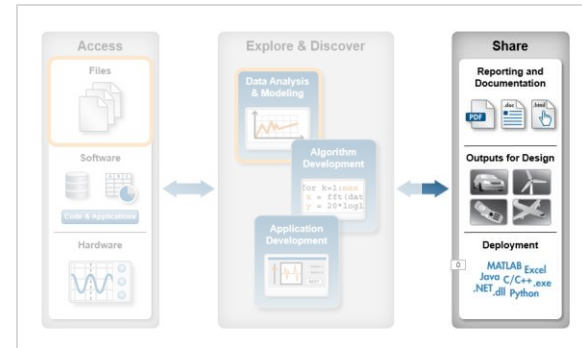
Deep Network Designer (Deep Learning Toolbox)



Classification Learner (Statistics and Machine Learning Toolbox)

# Document as you go – *your script is your report*

- Divide code into sections
- Embed outputs next to the code
- Add rich text formatting, equations, images, and hyperlinks
- Save directly to PDF, HTML, Word, and LaTeX



**Define Solar Correction by Location**

```

2 longitude= 42;
3 latitude = 44;
4 days = 1:365;
5 UTCoff = findOffset(timeZones,longi
6 eotCorr = equationOfTime(days);
7 solarCorr = 4*(longitude - 15*UTCof
8 delta = asind(sind(23.45)*sind(360*
9 sunrise = 12 - acosd(-tand(latitude
10 sunset = 12 + acosd(-tand(latitude

```

**Plot Sunrise and Sunset by Day of Year**

```

11 clf ; plot(days, sunrise, days, sun
12 axis([1 365 0 24])
13 title('Sunrise and Sunset')
14 xlabel('Day of Year') ; ylabel('Tim
15 hold on
16 patch([days flip(days)], [sunrise f

```

**Power Generation in Solar Cells**

**The Overall Approach**

In this example we will estimate the **power output** from a typical solar panel installation. We will use [12 noon on June 1st](#) in Boston to illustrate how to calculate the following:

- Solar time
- Solar declination and solar elevation
- Air mass and the solar radiation reaching the earth's surface
- Radiation on a solar panel given its position, tilt, and efficiency
- Power generated in a day and over the entire year

We will use these formulas to plot solar and panel radiation for our example day, and then plot the expected panel power generation over the course of a year. We'll use two MATLAB functions created for this analysis, `solarCorrection` and `hourlyPanelRadiation`, to streamline the analysis.

**Solar Time**

Power generation in a solar panel depends on how much solar radiation reaches the panel which in turn depends on the sun's position relative to the panel as the sun moves across the sky.

```

lambda = -71.06; % Boston Longitude
phi = 42.36; % Boston Latitude
UTCoff = -5; % Boston UTC offset
TZ = ['UTC' num2str(UTCoff)];
january1 = datetime(2016,1,1,'TimeZone',TZ); % January 1st
localTime = datetime(2016,6,1,12,0,0,'TimeZone',TZ) % Noon on June 1

localTime =
    01-Jun-2016 12:00:00

```

To calculate the sun's position for a given date and time we need to use *solar time*. Twelve noon solar time is defined to be the time when the sun is highest in the sky. To calculate solar time, we apply a correction to local time. That correction has two parts:

- A term which corrects for the difference between the observer's location and the local meridian
- An orbital term related to the eccentricity of the earth's orbit and its axial tilt

We'll use a MATLAB function created for this analysis called `solarCorrection`. For example, at noon on June 1, the solar time would be:

```

d = caldays(between(january1,localTime,'Day')); % Day of year
solarCorr = solarCorrection(d,lambda,UTCoff); % Correction to local time
solarTime = localTime + minutes(solarCorr)

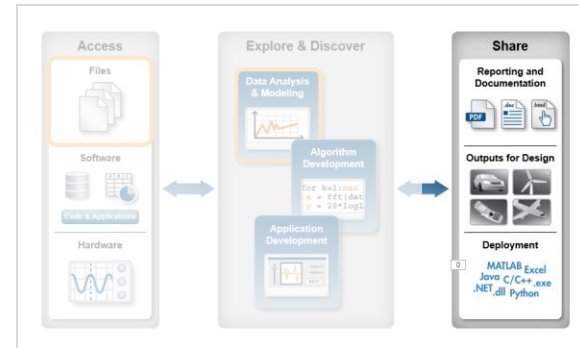
solarTime =
    01-Jun-2016 12:18:06

```



# Deploy and integrate MATLAB code

- Package and deploy MATLAB programs
- Generate code
  - C/C++
  - CUDA(Mex)
  - HDL



**MATLAB Coder**

The MATLAB Coder workflow generates standalone C and C++ code from MATLAB code. **To begin, select your entry-point function(s).**

Generate code for function:

---

**Finish Workflow**

**Source Code Generated Successfully**

You can now use the C code in your applications. [Learn more](#)

**Project Summary**

Functions kalmanfilter.m  
 Project Type MATLAB Coder  
 Project File kalmanfilter.prj

**Generated Output**

C Code C:\Work\codersdemo\_kalman\_filter\codegen\lib\kalmanfilter  
 Example main Files C:\Work\codersdemo\_kalman\_filter\codegen\lib\kalmanfilter\examples  
 Reports  Code Generation Report

# Start with low code ... and switch to code easily when needed

FlightData = 42584x12 timetable

|   | Time                 | FuelQuantity | Latitude | Longitude | OilPressure |
|---|----------------------|--------------|----------|-----------|-------------|
| 1 | 02-Jun-2001 05:50:04 | 7960         | 44       |           | 40          |
| 2 | 02-Jun-2001 05:50:05 | 7968         | 44       |           | 40          |
| 3 | 02-Jun-2001 05:50:05 | 7968         | 44       |           | 40          |
| 4 | 02-Jun-2001 05:50:05 | 7968         | 44       |           | 40          |
| 5 | 02-Jun-2001 05:50:05 | 7968         | 44       |           | 40          |
| 6 | 02-Jun-2001 05:50:06 | 7968         | 44       |           | 40          |
| 7 | 02-Jun-2001 05:50:06 | 7968         | 44       |           | 40          |
| 8 | 02-Jun-2001 05:50:06 | 7968         | 44       |           | 40          |
| 9 | 02-Jun-2001 05:50:06 | 7968         | 44       |           | 40          |

```
FlightData = sortrows(FlightData, "FuelQuantity", "descend");
FlightData = FlightData(FlightData.FuelQuantity >= 3863,:)
```

Update Code Copy

County General Ho... ▾

County General Hospital

VA Hospital

St. Mary's Medical Center

filename = C:\Work\data1.mat

50 -100 100

2

Interactive Controls

DATA IMPORT

Import Data

GRAPHICS

PREPROCESSING DATA

```
% Create plot of FlightData.Time and FlightData.FuelFlow
h2 = plot(FlightData.Time,FlightData.FuelFlow,"DisplayName","FuelFlow");

% Add xlabel, ylabel, title, and legend
xlabel("Time")
ylabel("FuelFlow")
title("FuelFlow vs. Time")
legend
```

Center and scale data

Preprocess Text Data

Smooth Data

TABLES AND TIMETABLES

Join Tables

Retime Timetable

Stack Table Variables

Synchronize Timetables

Unstack Table Variables

Live Tasks

MATLAB

```
function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData, responseData)
% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData,
% responseData)
% Returns a trained classifier and its accuracy. This code recreates the
% classification model trained in Classification Learner app. Use the
% generated code to automate training the same model with new data, or to
% learn how to programmatically train models.
%
% Input:
% trainingData: A matrix with the same number of columns and data type
% as the matrix imported into the app.
%
% responseData: A vector with the same data type as the vector
% imported into the app. The length of responseData and the number of
% rows of trainingData must be equal.
%
% Output:
% trainedClassifier: A struct containing the trained classifier. The
% struct contains various fields with information about the trained
% classifier.
%
% trainedClassifier.predictFcn: A function to make predictions on new
% data.
%
% validationAccuracy: A double representing the validation accuracy as
% a percentage. In the app, the Models pane displays the validation
% accuracy for each model.
%
% Auto-generated by MATLAB

% Extract predictors and response
inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2', 'column_3', 'column_4', 'column_5'});

predictorNames = {'column_1', 'column_2', 'column_3', 'column_4', 'column_5'};
predictors = inputTable(:, predictorNames);
response = responseData;
isCategoricalPredictor = [false, false, false, false, false];
classNames = {'Female', 'Male'};

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
classificationTree = fittree(...
predictors, ...
response, ...
'splitCriterion', 'gdi', ...
'MaxNumSplits', 100, ...
'Surrogate', 'off', ...
'classNames', classNames);

% Create the result struct with predict function
predictorExtFn = @(x) array2table(x, 'VariableNames', predictorNames);
```

untitled2\* x +

function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData, responseData)

% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData, % responseData)

% Returns a trained classifier and its accuracy. This code recreates the % classification model trained in Classification Learner app. Use the % generated code to automate training the same model with new data, or to % learn how to programmatically train models.

%

% Input:

% trainingData: A matrix with the same number of columns and data type % as the matrix imported into the app.

%

% responseData: A vector with the same data type as the vector % imported into the app. The length of responseData and the number of % rows of trainingData must be equal.

%

% Output:

% trainedClassifier: A struct containing the trained classifier. The % struct contains various fields with information about the trained % classifier.

%

% trainedClassifier.predictFcn: A function to make predictions on new % data.

%

% validationAccuracy: A double representing the validation accuracy as % a percentage. In the app, the Models pane displays the validation % accuracy for each model.

%

% Auto-generated by MATLAB

% Extract predictors and response

inputTable = array2table(trainingData, 'VariableNames', {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5'});

predictorNames = {'column\_1', 'column\_2', 'column\_3', 'column\_4', 'column\_5'};

predictors = inputTable(:, predictorNames);

response = responseData;

isCategoricalPredictor = [false, false, false, false, false];

classNames = {'Female', 'Male'};

% Train a classifier

% This code specifies all the classifier options and trains the classifier.

classificationTree = fittree(...

predictors, ...

response, ...

'splitCriterion', 'gdi', ...

'MaxNumSplits', 100, ...

'Surrogate', 'off', ...

'classNames', classNames);

% Create the result struct with predict function

predictorExtFn = @(x) array2table(x, 'VariableNames', predictorNames);

Apps

# MATLAB EXPO

Thank you



© 2023 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [mathworks.com/trademarks](https://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.



MathWorks ✓

@MathWorks

Share the EXPO experience

**#MATLABEXPO**

장규환 부장



pauljang@mathworks.com

